Natural Language Processing

Lab 9: Logistic Regression

Objective

The objective of this lab is to understand the logistic regression algorithm as a supervised classification method, particularly for tasks like sentiment analysis or text categorization in natural language processing. By following the theoretical steps, learners will grasp how to model binary or multinomial classification problems, compute probabilities using sigmoid or softmax functions, minimize loss through optimization, and incorporate regularization to improve generalization.

Theory

1. Feature Representation

For each input observation $x^{(i)}$, this will be a vector of features $[x_1, x_2, ..., x_n]$. We will generally refer to feature i for input $x^{(j)}$ as $x_i^{(j)}$, sometimes simplified as x_i, but we will also see the notation f_i, $f_i(x)$.

In traditional logistic regression, you design features like:

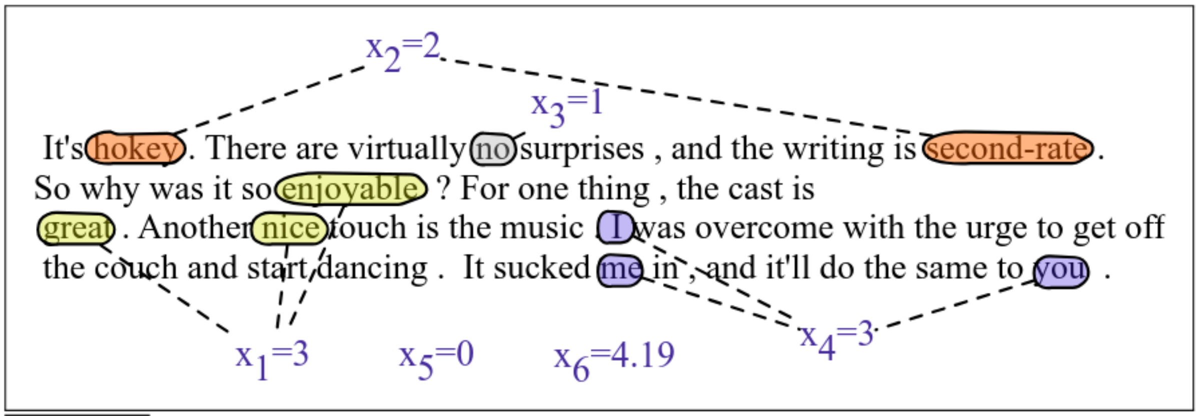
- Word counts
- N-gram presence
- Lexicon-based scores

With representation learning, these are replaced by dense vector representations learned automatically, such as:

- Word embeddings (Word2Vec, GloVe, FastText)
- Contextual embeddings (BERT, ELMo)
- Learned features from a neural network (CNN, RNN)

Var	Definition	Value in Fig. 5.2
x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	$\ln$(word count of doc)	$\ln(66) = 4.19$

Let's assume for the moment that we've already learned a real-valued weight for



x2=2
x3=1
It's hokey. There are virtually no surprises , and the writing is second-rate.
So why was it so enjoyable ? For one thing , the cast is
great . Another nice touch is the music I was overcome with the urge to get off
the couch and start dancing . It sucked me in , and it'll do the same to you .
x1=3
x5=0
x6=4.19
x4=3

Figure 5.2 A sample mini test document showing the extracted features in the vector x.

The classifier output y can be 1 (meaning the observation is a member of the class) or 0 (the observation is not a member of the class). We want to know the probability P(y=1|x) that this observation is a member of the class. So perhaps the decision is "positive sentiment" versus "negative sentiment", the features represent counts of words in a document, P(y=1|x) is the probability that the document has positive sentiment, and P(y=0|x) is the probability that the document has negative sentiment.

2. Sigmoid Function

Logistic regression solves this task by learning, from a training set, a vector of weights and a bias term. Each weight wi is a real number, and is associated with one of the input features x_i. The weight w_i represents how important that input feature is to the classification decision, and can be positive (providing evidence that the instance being classified belongs in the positive class) or negative (providing evidence that the instance being classified belongs in the negative class). Thus we might expect in a sentiment task the word awesome to have a high positive weight, and abysmal to have a very negative weight. The bias term, also called the intercept, is another real number that's added to the weighted inputs.

To make a decision on a test instance—after we've learned the weights in training—the classifier first multiplies each x_i by its weight w_i, sums up the weighted features, and adds the bias term b. The resulting single number z expresses the weighted sum of the evidence for the class.

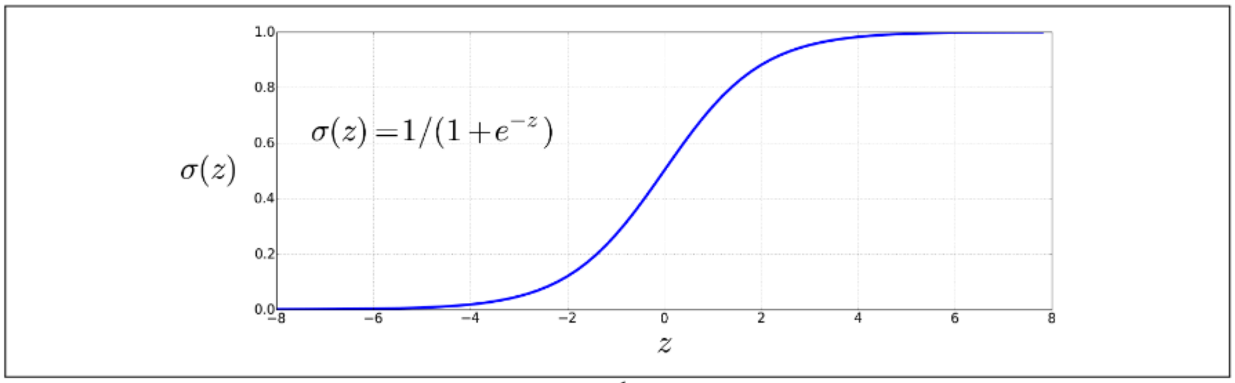
$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

Note that nothing forces z to be a legal probability, that is, to lie between 0 and 1. In fact, since weights are real-valued, the output might even be negative; z ranges from $-\infty$ to ∞.

To create a probability, we'll pass z through the sigmoid function, σ (z). The sigmoid function (named because it looks like an s) is also called the logistic function, and gives logistic regression its name.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$



If we apply the sigmoid to the sum of the weighted features, we get a number between 0 and 1. To make it a probability, we just need to make sure that the two cases, p(y = 1) and p(y = 0), sum to 1.

$$\begin{aligned} P(y=1) &= \sigma(\mathbf{w}\cdot\mathbf{x}+b) \\ &= \frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \end{aligned}$$

$$\begin{aligned} P(y=0) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b) \\ &= 1-\frac{1}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \\ &= \frac{\exp(-(\mathbf{w}\cdot\mathbf{x}+b))}{1+\exp(-(\mathbf{w}\cdot\mathbf{x}+b))} \end{aligned}$$

The sigmoid function has the property:

$$1-\sigma(x)=\sigma(-x)$$

so we could also have expressed P(y = 0) as:

$$\sigma(-(\mathbf{w}\cdot\mathbf{x}+b))$$

3. Decision Boundary

How do we make a decision about which class to apply to a test instance x? For a given x, we say yes if the probability P(y = 1|x) is more than .5, and no otherwise. We call .5 the decision boundary:

$$\text{decision}(x) \;=\; \begin{cases} 1 & \text{if } P(y=1|x)>0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{aligned}
p(+|x) = P(y=1|x) &= \sigma(\mathbf{w}\cdot\mathbf{x}+b)\\
&= \sigma([2.5,-5.0,-1.2,0.5,2.0,0.7]\cdot[3,2,1,3,0,4.19]+0.1)\\
&= \sigma(.833)\\
&= 0.70 \qquad (5.7)\\
p(-|x) = P(y=0|x) &= 1-\sigma(\mathbf{w}\cdot\mathbf{x}+b)\\
&= 0.30
\end{aligned}$$

4. Cross-Entropy Loss Function

The distance between the system output and the gold output is called the loss function or the cost function.

$$L(\hat{y},y) = \text{How much } \hat{y} \text{ differs from the true } y$$

$$p(y|x) = \hat{y}^{y}\,(1-\hat{y})^{1-y}$$

$$\begin{aligned}
\log p(y|x) &= \log\left[\hat{y}^{y}\,(1-\hat{y})^{1-y}\right]\\
&= y\log\hat{y}+(1-y)\log(1-\hat{y})
\end{aligned}$$

$$L_{\text{CE}}(\hat{y},y) = -\log p(y|x) = -[y\log\hat{y}+(1-y)\log(1-\hat{y})]$$

$$L_{\text{CE}}(\hat{y},y) = -[y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]$$

$$\begin{aligned}
L_{\text{CE}}(\hat{y},y) = &\; -[y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))]\\
= &\; -[\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)]\\
= &\; -\log(.70)\\
= &\; .36
\end{aligned}$$

$$
\begin{aligned}
L_{\text{CE}}(\hat{y},y) = & \quad -[y\log\sigma(\mathbf{w}\cdot\mathbf{x}+b)+(1-y)\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))] \\
= & \quad -[\log(1-\sigma(\mathbf{w}\cdot\mathbf{x}+b))] \\
= & \quad -\log(.30) \\
= & \quad 1.2
\end{aligned}
$$

5. Stochastic Gradient Descent

Our goal with gradient descent is to find the optimal weights: minimize the loss function we've defined for the model.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m}\sum_{i=1}^{m} L_{\text{CE}}(f(x^{(i)};\theta),y^{(i)})$$

$$w^{t+1} = w^t - \eta\frac{d}{dw}L(f(x;w),y)$$

$$\nabla L(f(x;\theta),y) = \begin{bmatrix} \frac{\partial}{\partial w_1}L(f(x;\theta),y) \\ \frac{\partial}{\partial w_2}L(f(x;\theta),y) \\ \vdots \\ \frac{\partial}{\partial w_n}L(f(x;\theta),y) \\ \frac{\partial}{\partial b}L(f(x;\theta),y) \end{bmatrix}$$

$$\frac{\partial L_{\text{CE}}(\hat{y},y)}{\partial \mathbf{w}_j} = -(y-\hat{y})\mathbf{x}_j$$

```
function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, $x$, $y$) returns $\theta$
    # where: L is the loss function
    #        f is a function parameterized by $\theta$
    #        x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
    #        y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$
repeat til done    # see caption
  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
    1. Optional (for reporting):            # How are we doing on this tuple?
       Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$       # What is our estimated output $\hat{y}$?
       Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$   # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
    2. $g \leftarrow \nabla_{\theta} L(f(x^{(i)}; \theta), y^{(i)})$         # How should we move $\theta$ to maximize loss?
    3. $\theta \leftarrow \theta - \eta g$                      # Go the other way instead
return $\theta$
```

Task 1

Create a logistic regression model for text classification without using any 3rd party libraries with the exception of spaCy's feature vector attribute. Take any labelled dataset from the internet. Bonus points for those who implement their own either handcrafted features or feature model.